THE DESIGN OF A USER INTERFACE FOR A COLOR, RASTER SCAN GRAPHICS DEVICE

Roger Lee Nesslage

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THE DESIGN OF A USER INTERFACE
FOR A
COLOR, RASTER SCAN GRAPHICS DEVICE

by

Roger Lee Nesslage

June 1976

Thesis Advisor: G. M. Raetz

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Design of a User Interface for a Color, Raster Scan Graphics Device		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1976 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Roger Lee Nesslage		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1976 13. NUMBER OF PAGES 98
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Graphics
Color
User-Interface
Raster
RAMTEK

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis is a summary of the design and implementation of a user interface for a color, raster-scan graphics display device. The problems, solutions and a general set of guidelines related to designing an interface for a color capable graphics device are discussed. The implementation of a software interface for a RAMTEK GX-100A with the PDP-11/50 computer is presented. The interface is implemented within the conventions of

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE
(Page 1) S/N 0102-014-6601 |

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (cont.)

the C programming language and executes under the UNIX operating system. Recommendations for further expansion of the interface are discussed.

The Design of a User Interface for a Color, Raster Scan Graphics Device

by

Roger Lee Nesslage
First Lieutenant, United States Marine Corps
B. A., Culver-Stockton College, 1971

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
JUNE, 1976

ABSTRACT

This thesis is a summary of the design and implementation of a user interface for a color, raster-scan graphics display device. The problems, solutions and a general set of guidelines related to designing an interface for a color capable graphics device are discussed. The implementation of a software interface for a RAMTEK GX-100A with the PDP-11/50 computer is presented. The interface is implemented within the conventions of the C programming language and executes under the UNIX operating system. Recommendations for further expansion of the interface are discussed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

This thesis is the design and implementation of a user interface for a color, raster-scan display device, specifically the RAMTEK GX-100A. The basic design and operation of the device as implemented in the Naval Postgraduate School Computer Laboratory is discussed.

The actual design and operation of the interface is presented as well as the problems which were considered in its design and construction. The main consideration in the design of the interface was how to effectively implement the use of color in a graphic display.

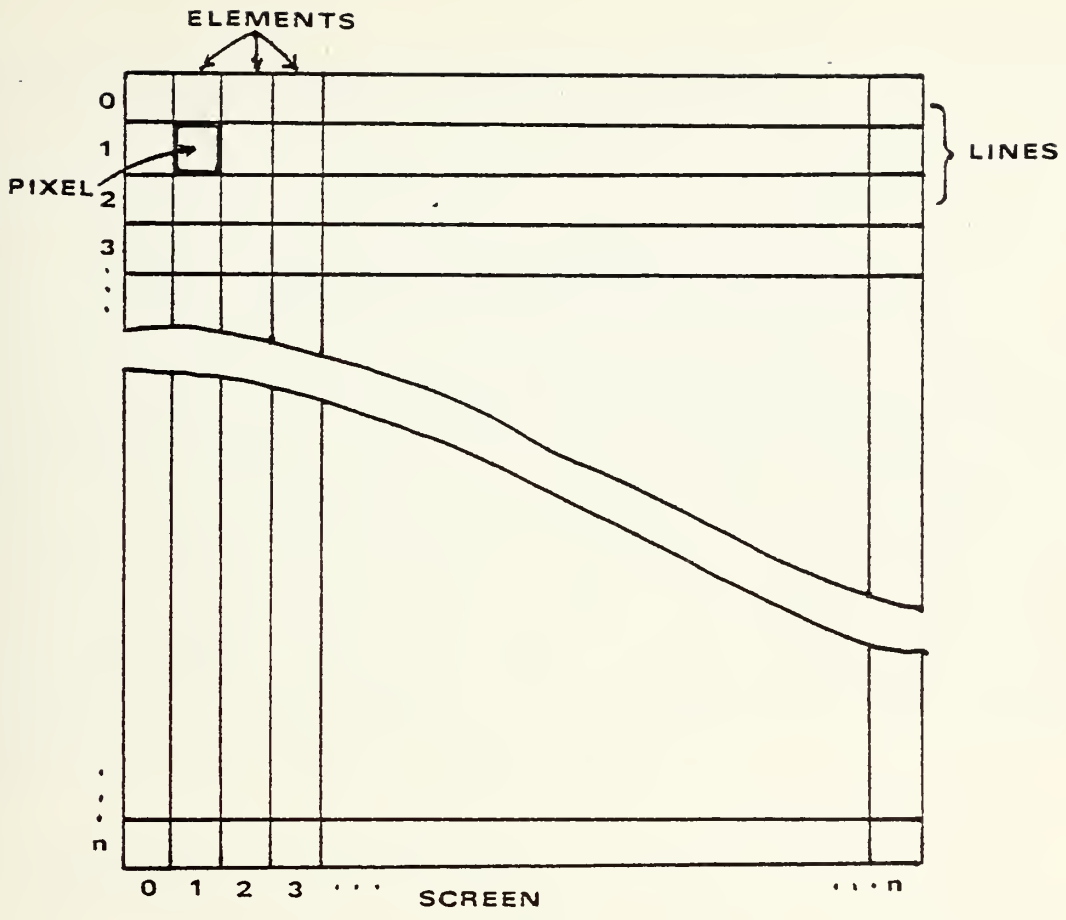
The concept of a color triple and color cube are introduced as natural extensions of the method used by color graphics devices. However, the triple and cube are strictly defined by limitations in the hardware that restrict the number of colors that can be generated to 4096.

A user's manual was written to enable the user to utilize the interface from a program written in the high level language, C [1], supported by the UNIX operating system on the PDP-11/50. It is included in Appendix A.

II. GENERAL DESCRIPTION OF RASTER SCAN DEVICES

The cathode ray tube is the most widely used graphics display device today. It consists basically of an electron gun which projects a beam of electrons onto a screen which has a phosphor coating. The phosphor is excited when struck by the beam of electrons and emits visible light which is perceived by the viewer. Due to the short persistence of the phosphor utilized on most cathode ray tubes, it is necessary to repeatedly excite it to minimize fading of the image and reduce visible flicker. There are two basic methods used to refresh a display. The first is to directly trace the image being produced several times a second with the electron beam turned on. The second is to scan the entire screen with the beam, while only turning the beam on and off when desired to produce the image. The second is known as the raster scan technique. It is the same technique used in commercial television sets.

In raster scan devices, the screen is divided into lines and elements. Lines extend horizontally across the screen and elements are discrete positions on a line. A specific element on a specific line is called a pixel. Figure 1 illustrates the relationship of lines, pixels and elements to each other and the screen. The number of pixels determines the resolution of the screen. The more pixels, the higher the resolution of the picture.



Line, Element, Pixel Relationshio

Figure 1

The added feature of color in a raster scan device requires the addition of two more electron guns and a more complicated phosphor coating on the screen. The phosphors, representing the primary colors of blue, green and red are located in each pixel area and are called a triad. The three guns are used to produce three electron beams to scan the screen. These beams are independently varied in intensity to excite any one, two or three phosphors to produce the desired color.

In order for an image to be drawn in color, not only must the electron guns be turned on and off at the proper locations, but the combination of beams and the intensity of

each must be adjusted to produce the color.

III. THE RAMTEK GX-100A GRAPHICS DISPLAY

The RAMTEK is an interactive, raster-scan, color graphics display device. It consists of a bi-directional interface, a display generator, solid state memory, a video look-up table, device multiplexor, keyboard, cursor generator, video generator, and a cathode ray tube [2]. Figure 2 illustrates the relationship of the hardware and the component parts' relationships with each other.

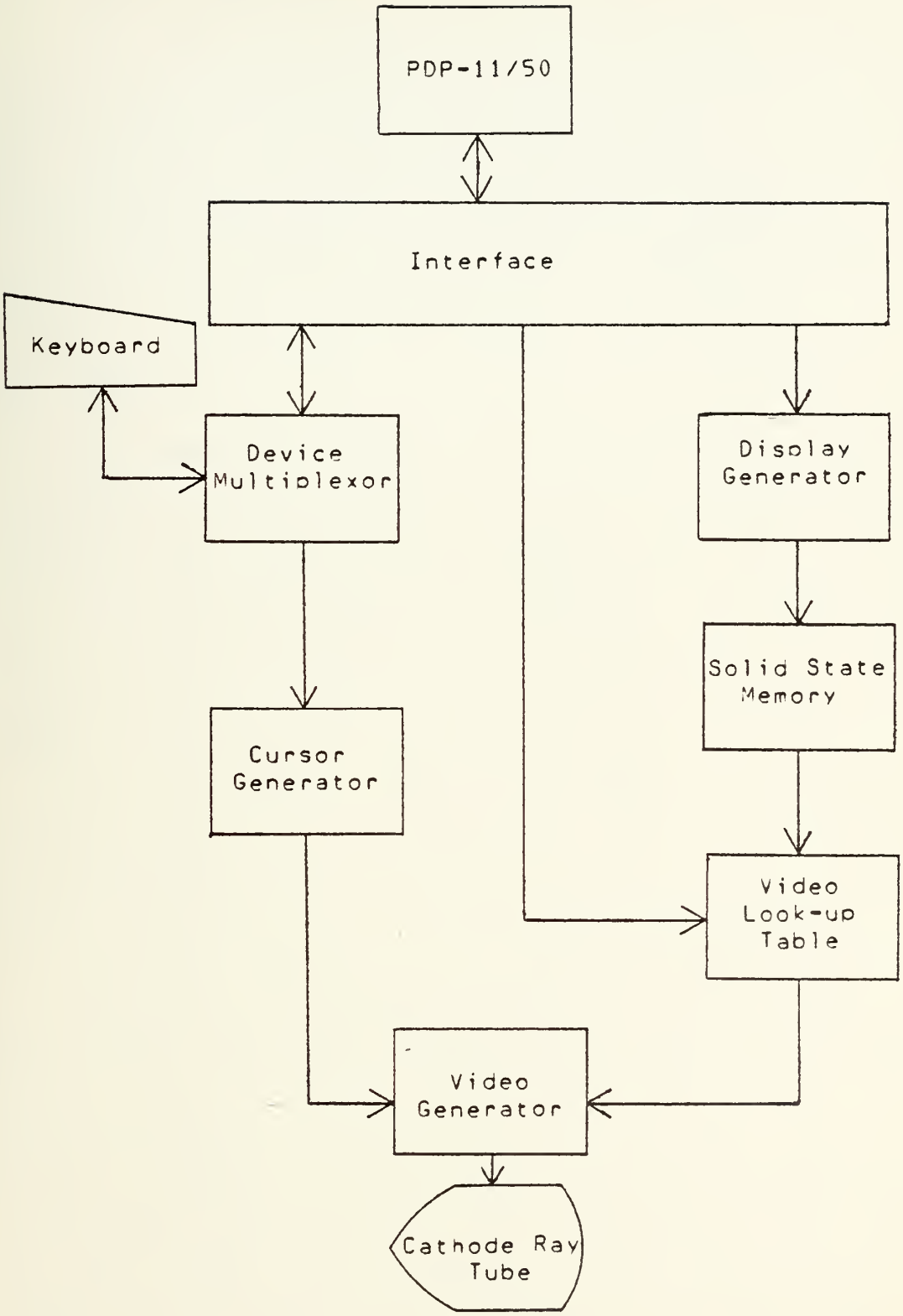
A. BI-DIRECTIONAL INTERFACE

The bi-directional interface connects the PDP-11 with the video lookup table, display generator and the device multiplexor. The interface establishes data paths to and from the CPU and the device multiplexor, the video lookup table, and the display generator.

B. DEVICE MULTIPLEXOR

The device multiplexor interconnects with the cursor generator and the keyboard. It enables the CPU to receive data via the RAMTEK keyboard.

The keyboard converts the keystrokes into extended ASCII character codes which are transmitted to the device multiplexor.



RAMTEK Hardware Configuration

Figure 2

C. DISPLAY GENERATOR

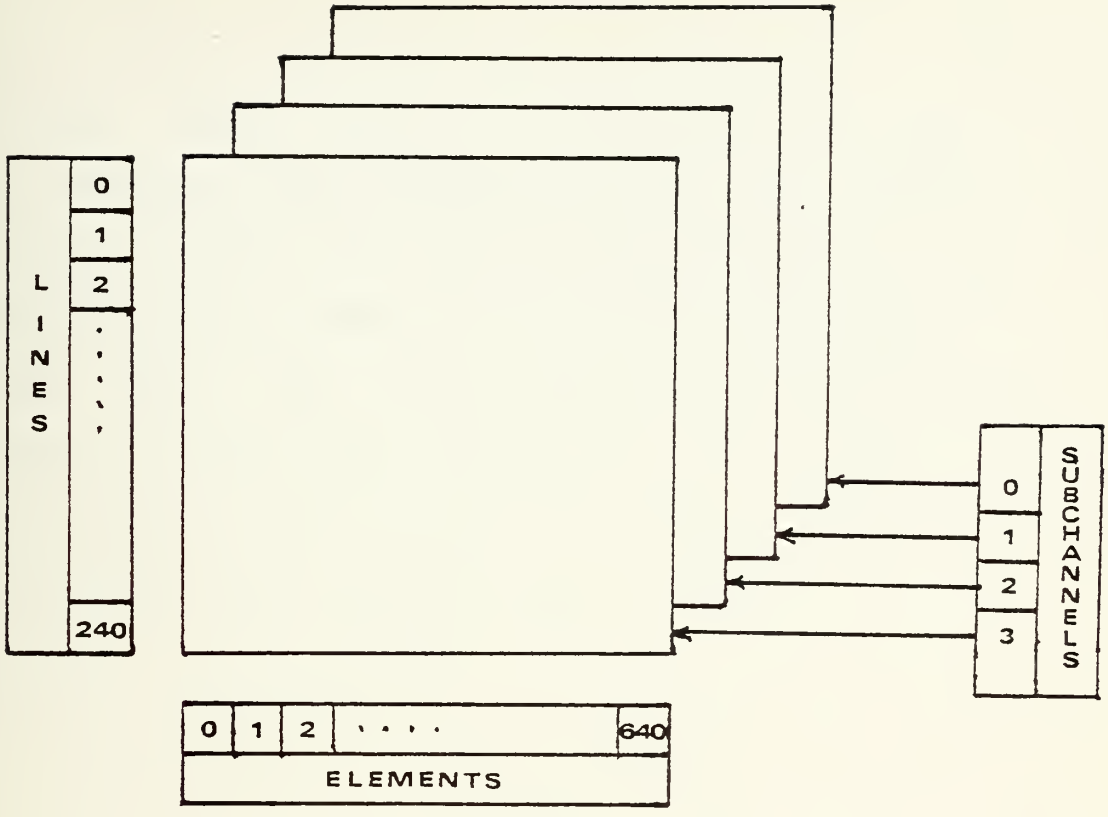
The display generator contains the standard ASCII character codes and hardware logic such that dot-matrix character patterns are generated and passed to the memory. It also takes input data and processes it according to the instructions received and writes the appropriate data into the memory so the proper image will be displayed.

D. MEMORY

The organization of the solid state refresh memory is the key to the operation of the RAMTEK. The memory consists of up to eight memory boards (planes) which have the capacity to store 512 lines by 640 elements. The RAMTEK used in this work has four memory planes and an effective address space of 240 lines by 640 elements.

As illustrated by figure 3, the memory can be viewed as a three dimensional structure with the x-axis corresponding to the elements, the y-axis corresponding to the lines and the z-axis corresponding to the display subchannels.

The display subchannels represent a binary pointer into the video lookup table by only permitting certain planes of memory to be written in. Thus the binary code on a certain z-axis represents the pointer into the video lookup table. This establishes the color to be displayed at that location on the screen. For example, if the z-axis at line five, element 56 contains a 0101, the pointer into the video look-up table points to entry five.



Memory Configuration

Figure 3

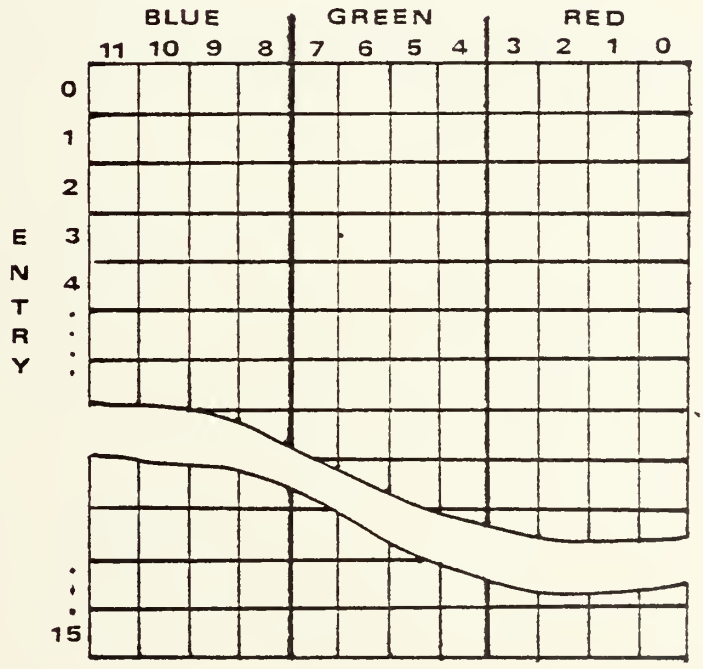
E. VIDEO GENERATOR

The video generator reads the information stored in the refresh memory and generates the composite signals for the cathode ray tube (CRT) to display the image on the screen.

F. VIDEO LOOKUP TABLE

The video lookup table is the center of the color generating capability of the RAMTEK. It contains sixteen arbitrary colors from address zero to fifteen in the table. As described above, the combination of memory planes enabled by the selected subchannels represents the pointer to an entry in the video lookup table. Each entry in the video

lookup table consists of a twelve-bit code broken into three four-bit binary fractions which correspond to intensities of blue, green and red. As illustrated in figure 4, bits zero through three represent red, four through seven represent green and eight through eleven represent blue. Therefore, any of 4096 colors can be represented in the twelve bits. Only sixteen colors can be displayed on the screen at any one time because having four planes of memory to represent the pointer to the video lookup table limits the pointer value to zero through fifteen.



Video Lookup Table

Figure 4

IV. POSITIONAL ADDRESSING

There are three methods of addressing the RAMTEK screen; absolute, indexed and relative. As stated earlier, addressing a raster scan device is accomplished by specifing the line and element number which indicate the address of a pixel on the screen.

Because of the low line resolution of the device used, only even lines are addressable. Therefore the line addressing is from zero to 480 where zero and one address the same line, two and three address the same line and so on. The address that is currently being referred to is known as the current operating point (COP). It is like a pointer to a location on the screen where the next entity will be displayed.

A. ABSOLUTE ADDRESSING

By specifying a line and element number, a single pixel is addressed in the same way as a point is addressed in a cartesian coordinate system. The COP is at that pixel.

B. INDEXED ADDRESSING

The RAMTEK provides two index registers which can be used in addressing the RAMTEK screen. To address a location using index registers, the specified line and element numbers are summed with the contents of the line and element index registers to yield the address of a pixel on the screen. For example, if the specified address is line 100, element 200 and the line and element index registers contain

26 and 30 respectively, the resulting address (COP) is line 126, element 130.

C. RELATIVE ADDRESSING

This method takes the specified line and element numbers and considers them to be displacements, that is, the number of lines and elements to add to the COP address to yield a new COP. For example, if the specified displacements are 20 lines and 30 elements, and the COP is line 100, element 140; the new COP is line 120, element 170.

V. CONTROL MODES

The RAMTEK operates in any one of eight control modes [2]. Each control mode can be modified by selected control flags. The modes and their applicable control flags are summarized in Tables I and II. More detailed descriptions of the modes and how they work are located in Appendix A.

TABLE I

CONTROL MODE DEFINITIONS

Mode	Functional Description
Alphanumeric Data	Writes alphanumeric characters on the screen.
Transverse Data	Decodes the data as a single bit per pixel and writes the data in an eight pixel wide vertical column from top to bottom.
Raster Data	Same as Transverse data but writes left to right on the same line.
Complex Data	Decodes the data as four bits per pixel and writes left to right on the same line.
Graphic Vector	Draws continuous lines between arbitrary end points.
Graphic Plot	Draws a histogram style plot by drawing continuous lines between successive y-axis points while incrementing the x-axis from left to right.
Graphic Cartesian	Draws solid rectangles between arbitrary end points.
Graphic Element	Draws a dot at specified point.

TABLE II

CONTROL FLAG DEFINITIONS

Name	Functional Description
Index Addressing	Causes indexed addressing to be in effect.
Reverse Background	Causes ones written in memory to become zeros and vice-versa.
Additive Write	Causes the new image to written "on top" of the existing image.
Double Width	Doubles the width of generated data by writing each bit of data into two elements instead of one.
Fixed Point	Uses the point defined by the contents of the line and element index registers as the starting point for each graphic entity.

VI. SPECIAL FUNCTIONS

There are two special functions which alter the display. The first is the erase instruction and the second is the scroll function.

The erase function simply writes all zeros or ones into memory depending on the state of the background flag in the control mode and the subchannels selected.

The scroll function creates a moving window effect by moving the image vertically for a specified number of lines. A positive or negative number indicates which direction the image will move. Images that are scrolled off the top of the screen re-enter the bottom and vice-versa.

VII. GENERAL USER INTERFACE DESIGN

One of the governing constraints in developing software for graphics output is the fact that it is harder to describe a picture than it is to describe a table of words or numbers.

For this reason, some of the normal principles of good system design apply with exceptional force. The user who has the job of describing a picture must be provided with a set of instructions and graphic tools that enable him to concentrate on the image he wishes to create. The user must not be distracted by irritating details imposed by the peculiarities of the hardware. He must be allowed to express algorithims as he conceives them, and be able to trace and correct errors with a minimum of trouble. It is essential that the graphics system be simple to program [3].

An excellent general approach to designing a user interface for graphics systems is expressed by Wagner and LaHood [3]. It describes some general guidelines in which a level structure is built.

A. MULTILEVEL STRUCTURE

A multilevel design is based on building blocks which, at the lower levels, involve efforts to provide an efficient interface between the computer and the graphic device. Higher levels of software are then developed to increase the productivity of the total system. Routines on the lower levels are dependent on hardware, but as levels increase, the dependence on hardware decreases, and the complexity of

the functions performed by software increases. At the highest level the routines are totally independent of hardware.

1. Level I - Primitive Hardware Oriented Subroutines

Level I routines are intimately concerned with the hardware characteristics of the device. These form the basis for building Level II routines. Examples are primitive display command utilities, display instruction assembler, interrupt handling routines, and device controllers.

2. Level II - Complex Information Handling Subroutines

Level II routines combine the facilities of the operating system with the hardware oriented routines of Level I to provide the programs and user oriented languages to be used at Level III. Examples are file management, buffer management utilities, and utilities that convert user defined coordinate systems to a device dependent coordinate system.

3. Level III - Generalized User Oriented Subroutines

Level III routines have the ability to perform several Level II routines to accomplish a single task. The routines allow specification of display functions through device independent parameters. They are called within a procedural language like C [1], Algol or Fortran. Therefore, they add to the capabilities of an existing implementation. Examples are plot routines, automatic scaling, print and label routines.

4. Level IV - Application Language Processors

Level IV programming support is directed at the user

who is not a programmer. It is intended to make him less dependent on programming by simplifying the task of specifying graphic output. Once implemented by using routines in Levels I, II and III, and other programming techniques, user languages provide access to the graphic device and increase its usefulness. Examples of some Level IV user-oriented, high level language systems are MIT Sketch Pad [4], Bell Animation Language [5,6] and GM Design Analysis System [7].

VIII. USER INTERFACE DESIGN FOR THE RAMTEK GX-100A

A. GENERAL

In the design of a user interface for a color graphics device a primary consideration is to make the use of color as easy for the user as possible. The user should be permitted to make up colors at the device and be given a simple method of selecting the colors desired in the application program. The user should be allowed to modify the colors at will either during program execution or by a pre-execution selection process. The user should be relieved of selecting the exact bit patterns needed to produce the color on the screen. Elimination of device dependent programming allows the user to think of color as it applies to a problem.

It is necessary to isolate the user from the low-level assembly languages to allow concentration on the problem and not have to worry about when and how to issue the proper hardware instructions to the display device. A user should

be able to think in terms of a virtual screen space of his own definition and not be concerned with hardware addressing.

When a user interface is designed, it should be easily integrable with the host computer language. The easiest way to accomplish this is to use subroutine calls from the user program. The calls should be kept as simple to use as practicable. Toward this end, the number of parameters involved should be kept to a minimum.

The user should be protected from abnormal termination as much as possible by the interface. As an example, the interfacing routines can check the passed parameters to ensure that they are correct. They should indicate errors and what they are to simplify debugging. Graphic tools provided to the user should encourage concise, clear programming.

The overall principles and methods of level design as discussed earlier were followed in implementing the interface for the RAMTEK. Since Level I routines were essentially complete or were hardware implemented, the resulting design took place more in the Level II and Level III areas. It was decided to implement the interface as a set of subroutine calls from the C [1] programming language because C is the primary language on the host computer and because of its simple subroutine calling format.

As a first step, a set of primitives were designed to form the basis for other routines to build on.

1. Primitive Functions

The primitive functions make up the majority of the level II and level III routines. The criteria followed in choosing the primitive functions were clarity, convenience, and compactness. Clarity means primitives are to perform only one function and that function must be clear and simple.

Due to the hardware capabilites of the RAMTEK, several primitives became user routines on Level III as well as being primitives. The construction of a set of primitives condensed the entire capability of the RAMTEK hardware functions to 21 instructions or subroutines. The primitives which are applicable to user programs are described in Appendix B.

B. APPLICATION OF COLOR

After the design of the primitive set, it was necessary to design a method by which the user would be able to easily utilize and understand how to get color into a display and control or modify it. Two key concepts were used. The first was extended from an actual hardware feature. It was the video lookup table concept translated into a color table. The available colors can be thought of as a book of colors from which the user can choose the page he wishes to use. The page corresponds to the color table. All the user needs to do is select the page (color table) and color (entry) desired and then draw the entities using that color. The concept works well, but a problem arose as to how to present the information to the user so it is understandable

and useable. Humans have the characteristic of not all perceiving the same color the same way. A shade of blue to one person, may not be the same to another. And of course, some humans have varying degrees of color-blindness. Also, reproduction of color in manuals and photographs may not be accurate representations of the color the RAMTEK displays on its CRT. As a result of these considerations, it was decided that an interactive program should be designed and implementated into a subroutine call to allow the user to examine and work with the colors while actually viewing them on the RAMTEK screen.

1. Interactive Program

The interactive subroutine's purpose is to allow the user to see and modify color tables while seated at the RAMTEK keyboard. It embodies several features.

a. Self-tutorial

As the user utilizes the routine, instructions are given at each step as to how to progress through the various editing modes.

b. Color Table Display

The page representing a specific color table is designed so the user sees the fifteen colors in three modes; a solid block of color, a line of color and alphanumeric characters of the color are displayed. The display thereby shows the different appearance color can take on when displayed in different shapes.

c. Multiple Edit Modes

The routine provides five methods of editing and

modifying color tables and their entries. This gives maximum flexibility to the novice as well as the more advanced user. The use of the interactive routine is explained in detail in Appendix A.

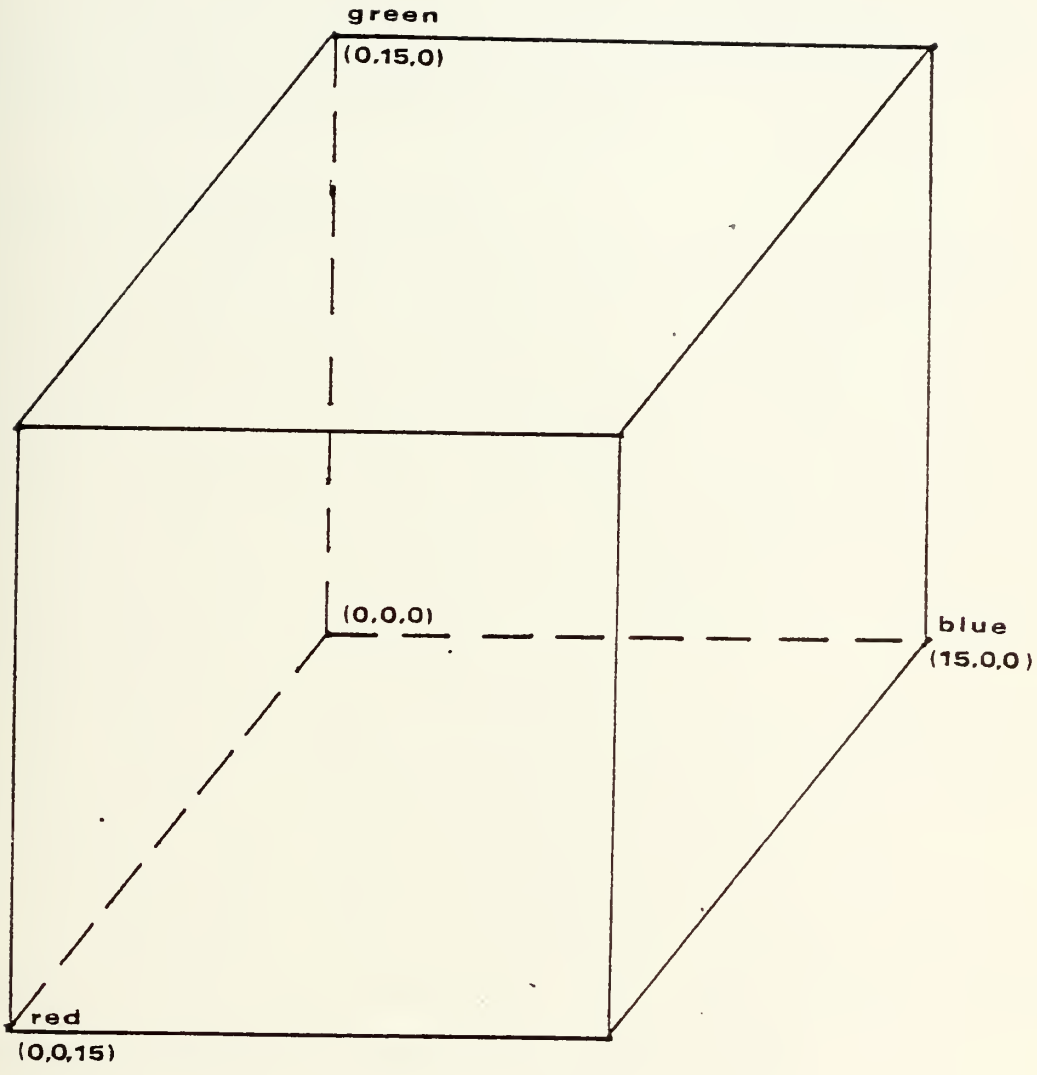
2. Color Cube

The color cube transforms the video lookup table into a third dimension to further aid in communicating to the user how to construct colors and how they relate to each other. The concept of the color cube, as illustrated in figure 5, describes the 4096 colors that can be created by defining the colors as ordered triples as follows: (0,0,0) to (0,0,15) represent shades of red, (0,0,0) to (0,15,0) represent shades of green and (0,0,0) to (15,0,0) represent shades of blue. By a careful inverting and combining of the three axis, many of the 4096 colors can be created.

C. VIRTUAL DISPLAY SCREEN

One of the important considerations of a successful user interface was met by writing a routine to allow the user to define the address space of the screen. The user defines the virtual screen by giving floating point cartesian coordinates representing the lower left and upper right corners of the screen. The interface remembers his declaration and, until the user changes the virtual screen, all addresses are transformed from the virtual coordinates given into real screen coordinates. The process is completely transparent to the user. To further aid the viewer, an axis system, centered where desired, can be displayed on the screen.

If the user chooses not to declare a virtual screen, the



Color Cube

Figure 5

interface defaults to an 100 by 100 unit screen. This isolates the user from real screen addressing problems.

All virtual screen displacements are also scaled into real screen displacements, thereby giving the user total freedom to define the virtual screen as long as a standard cartesian coordinate system is used.

D. HIGH LEVEL GRAPHIC TOOLS

A set of level III graphic tools was also designed, and its use and descriotion is detailed in Appendix A. Here simplicity, usablity and understandablity were the driving factors in the design. When possible, the user never has to pass more than four parameters. The higher level tools are logical operations that require the programmer to remember as little as possible when utilized.

All high level tools which draw graphics figures scissor [8] the image by permitting only the portion of the image that falls in the virtual screen to be displayed.

All routines return an indication if an error has occured or an incorrect parameter has been passed, thereby providing a debugging capability to the user.

E. INPUT AND OUTPUT ROUTINES

Three subroutine calls were developed to provide the user with an interactive input/output capability. The first is used by the user to retrieve single ASCII character codes from the RAMTEK keyboard. The second is used by the user to return an octal or decimal number from the RAMTEK keyboard. The third is used to return a floating point number to the

user from the RAMTEK keyboard. These routines provide a powerful tool to the user in programming.

F. MAINTAINABILITY

The functional design and level structuring serve to make servicing the interface an easy task. When a specific function does not perform as expected, the portion of the interface which applies to that function is easy to find and isolate for testing and debugging. Most of the interface routines do error checking and self-documentation when an error occurs.

IX. RECOMMENDATIONS

A. USER INTERFACE ADDITIONS

The following modifications to the user interface are recommended. The design and implementation of a complementing data structure to hold and control display entities for transfer to and from the RAMTEK on user command would be an excellent and powerful addition to the present interface. The data structure could utilize the existing routines with few changes. Such an addition would aid the user by allowing selective display and erasure of user entities with a single command.

Also, further interfacing of the RAMTEK keyboard to provide cursor control and readback of virtual or real screen coordinates from its location would be extremely useful. Several of the unused keys could be turned into function-type switches as well.

B. ADDITION OF HIGH LEVEL USER ROUTINES

More routines to give the RAMTEK a dynamic graphics capability is possible, but limited by the raster rate and data transfer rate. Move routines and selective erasure of entities can be accomplished within the confines of a well defined data structure.

X. CONCLUSIONS

The user interface as designed and implemented is working with no known software bugs. The entire range of hardware capabilities of the RAMTEK have been successfully incorporated and tested. Several successful high level routines, most notably the plotting routines, demonstrate a high degree of usefulness and are indicative of the capabilities that a color device in conjuction with an adequate software interface has to offer.

APPENDIX A: USER MANUAL

I. INTRODUCTION

The RAMTEK GX-100A is a color, raster scan display device, the heart of which is a color cathode ray tube not unlike a home color television set. In fact, its raster scan method for generating a picture is identical to commercial cathode ray tubes. At this point the similarity ends. Instead of decoding signals transmitted by a television broadcast station, the RAMTEK gets the information for its picture from a special video generator which reads the contents of a MOS refresh memory which contains all the information needed to produce an image on the screen. It is not within the scope of this manual to go into the details of how the electronics functions, but to inform the user on how to use the device to display the desired image.

A brief description of the parts of the RAMTEK will be given followed by a discussion of its various display modes. Next a discussion of how the user can define and utilize colors will be presented.

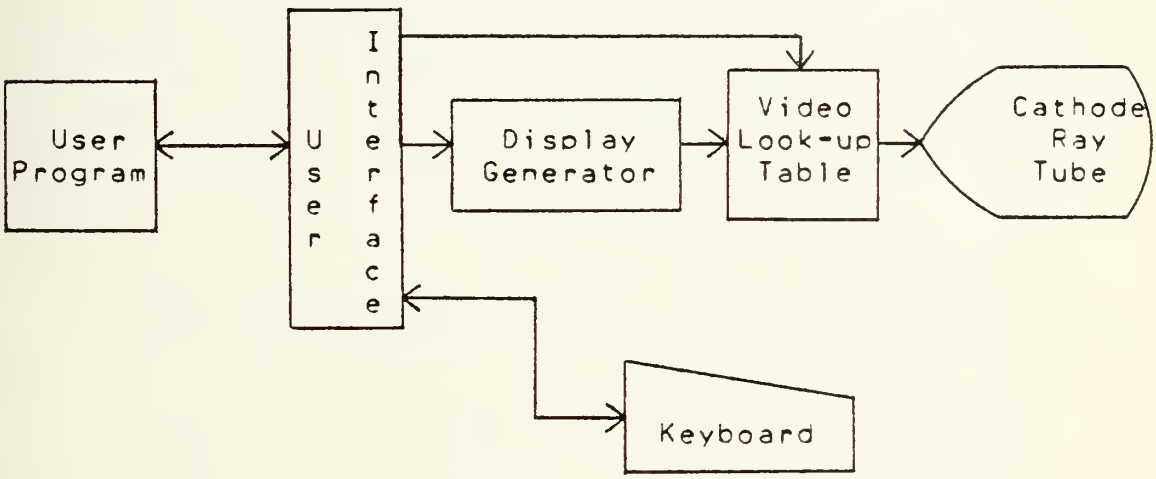
The user is also directed to Appendix B for information on how the user routines implemented in the interface are used and what their functions are. Periodically small sequences of C code will be listed. These bits of code are not intended to run without the necessary C and user interface routines that must precede them. They are presented to demonstrate how certain instructions are utilized.

II. THE RAMTEK DISPLAY SYSTEM

A more detailed discussion of the functioning of the RAMTEK can be found in the RAMTEK GX-100A [9] and RAMTEK GX-100B [2] programmers manuals.

A. GENERAL

The RAMTEK consists of a software interface to the user, a display generator, keyboard, video lookup table and a cathode ray tube. Their relationship is illustrated in figure A-1.



User Interface, RAMTEK Relationship

Figure A-1

The user program executes on the PDP-11/50. When the appropriate instruction is executed, the interface between the program and the display generator for the RAMTEK is activated. Information can be sent via the interface to the

display generator from the user program or the keyboard. Information can also be sent from the keyboard via the interface to the user program. After the display generator interprets its instructions and constructs the display in its ROM memory, the image is generated on the CRT according to the colors contained in the video lookup table. Any image will remain, unless over-written, for an indefinite period of time in the ROM memory.

The keyboard is interfaced to the user program and to the instruction display generator. Information can be passed from the keyboard in the form of ASCII character codes to the user program or to the display generator and finally to the CRT screen. The keyboard also contains eight function switches to control the action of a cursor which is displayed on the screen. The address of the cursor on the screen is always available to the user program via the interface.

The video lookup table contains the coded information which is used in generating color. The table is available via the interface to the user program. By modifying the contents of the table, various colors can be displayed on the screen.

The RAMTEK graphics display in the Naval Postgraduate School Computer Laboratory utilizes a raster scan cathode ray tube with a screen resolution of 240 horizontal lines and 640 elements on each line. A specific element and line is referred to as a pixel. This device has a high element resolution and a low line resolution. Therefore, the

quality of some images on the RAMTEK can be rather poor. Lines that are not horizontal or vertical are drawn with a noncontinuous, staircase effect. When utilizing the device these resolution factors should be kept in mind especially when operating in a data mode. Addressing the CRT screen is discussed in the section on Virtual Screen Addressing.

Color is generated by mixing three primary colors; blue, green and red in varying intensities. Using color is discussed in a later section.

B. VIRTUAL SCREEN ADDRESSING

The initialization of the RAMTEK, which is accomplished by the instruction ramtek(), presets the virtual screen to a standard cartesian coordinate address space with (0.0,0.0) as the lower left corner of the screen and (100.0,100.0) as the upper right corner. The virtual screen can be redefined by issuing a screen(x,y,x1,y1) instruction.

screen() redefines the virtual screen by setting the lower left corner to (x,y) and the upper right corner to (x1,y1). The only restrictions are that the two points define a valid cartesian coordinate system and the four values are floating point numbers. All addresses and displacements issued subsequent to a screen() instruction are interpreted according to the defined virtual screen.

The virtual screen is addressed by specifing an x,y coordinate with a strtxy(x,y) instruction. The specified address (x,y) establishes the current operating point (COP) at that address. The COP is initially located at the lower left corner of the virtual screen whenever a screen is

defined. The COP is occasionally modified after certain operations are performed as indicated in Appendix B and in the section on display modes.

There are three modes of addressing the screen; absolute, indexed and relative.

1. Absolute Addressing

In absolute addressing the user specifies a specific location on the user-defined screen by issuing a strtxy(x,y). The 'x' and 'y' values establish the location of the COP.

2. Indexed Addressing

To utilize indexed addressing, the user must issue an index(i,x,y) instruction. If 'i' is equal to one, indexed addressing is initiated. Subsequent to this instruction, all addresses specified by the user are interpreted according to the values 'x' and 'y' contained in the last index() instruction. For example, if a strtxy(10.0,10.0) is issued after an index(1,15.0,20.0) has been issued, the resulting COP is (25.0,30.0).

To deselect indexed addressing it is necessary to issue an index() instruction with 'i' equal to zero. The x and y values, if any, are ignored and the user is returned to the absolute addressing mode.

3. Relative Addressing

The hardware relative addressing is implemented by a single instruction. The issuing of a pointr(x,y) instruction utilizes relative addressing. When issued, the COP is calculated by taking the old COP x and y values and summing

them with the pointr() x and y values. For example, if the old COP is (22.0,50.0), and a pointr(30.0,-10.0) is issued, the new COP becomes (52.0,40.0). Relative addressing is only utilized when a pointr() instruction is issued and always refers to the last COP.

III. CONTROL MODES AND CONTROL FLAGS

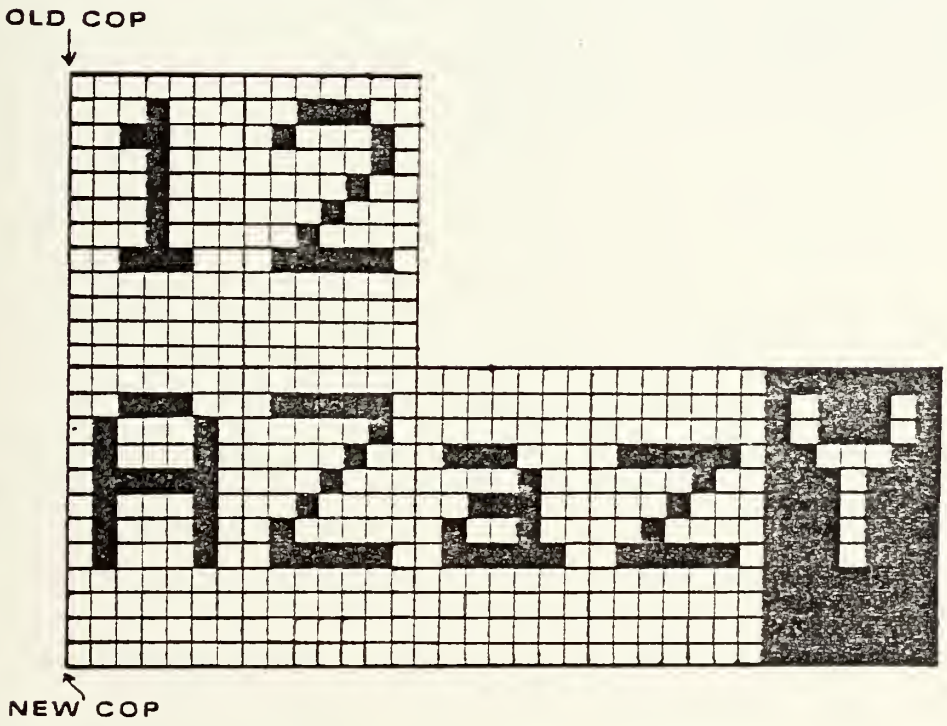
The RAMTEK operates in any one of eight modes. Each mode has associated with it certain control flags which modify its operation in the mode selected. Table A-I summarizes the control modes and control flags and their relationship.

Control modes are set by issuing a setmode(a,b) instruction. The value of 'a' indicates the mode the RAMTEK is to be placed in. The value of 'b' tells the interface whether to save the control flags of the previous mode or not. The values of 'a' and 'b' that may legally be assigned are summarized in Appendix B. The control modes are as follows:

A. ALPHANUMERIC DATA MODE

Alphanumeric Mode is the default mode upon initialization of the RAMTEK. All data() instructions issued while in this mode are interpreted as passing alphanumeric data. The data is processed leftmost byte first and is interpreted as eight bit extended ASCII character codes. Each received character code is transformed into a dot matrix character font which is five real screen elements by seven real screen

lines in dimension. The character font is within a seven element by twelve line matrix as illustrated in figure A-2. The entire character set is illustrated in reference [2].



RAMTEK Character Font

Figure A-2

TABLE A-I

CONTROL MODES AND CONTROL FLAGS

Definitions:

IX - Indexed Addressing
BK - Reverse Background
AW - Additive Write
DW - Double Width
FP - Fixed Point
YES - Indicates the Control Flag has effect in the Control Mode
NO - Indicates the Control Flag does not have effect in the Control Mode

Mode		Control Flags				
No.	Name	IX	BK	AW	DW	FP
0	Alphanumeric	YES	YES	YES	YES	NO
1	Transverse Data	YES	YES	YES	YES	NO
2	Raster Data	YES	YES	YES	YES	NO
3	Complex Data	YES	YES	NO	YES	NO
4	Grpahic Vector	YES	YES	NO	NO	YES
5	Graphic Plot	YES	YES	NO	YES	YES
6	Graphic Cartesian	YES	YES	NO	NO	YES
7	Graphic Element	YES	YES	NO	YES	NO

The transmitted ASCII characters are displayed starting at the COP from left to right on the same line as illustrated in figure A-2. If more than 91 characters are written on one line, they will "wrap around" and overwrite on the same line. No more than twenty character lines can be displayed on the screen.

Completion of execution of a data() instruction causes a "line feed" and "carriage return". Each subsequent data() instruction will display on the next alphanumeric line at the same starting element as illustrated in figure A-2. The new COP after each execution is also illustrated in figure A-2.

When the characters are drawn on the RAMTEK screen, the character appears in the designated color while the rest of the matrix is drawn in the background color.

1. Applicable Control Flags

a. Indexed Addressing Flag

The effect of indexed addressing is as explained above in Virtual Screen Addressing.

b. Reverse Background Flag

If set, the character is displayed in the background color and the rest of the matrix in the selected color as illustrated by the 'Y' in figure A-2.

c. Additive Write Flag

If set, the character is written over whatever image happens to be at the same location on the screen. The use of the additive write flag is convenient for combining characters to create special characters or writing

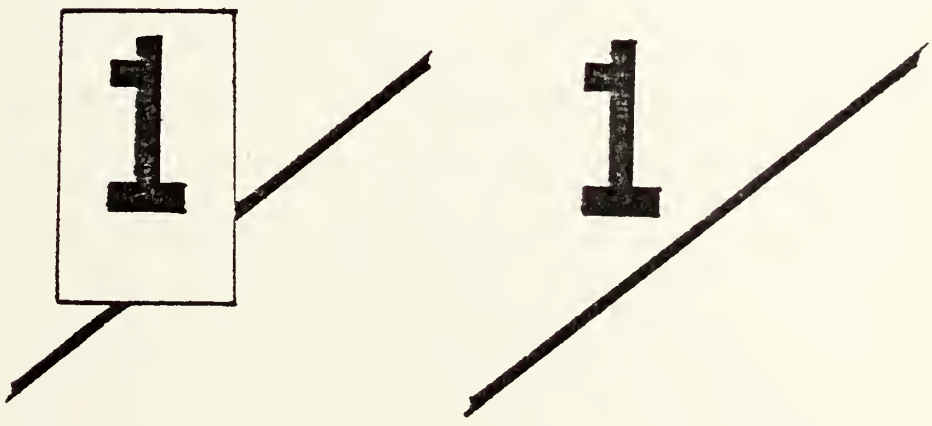
characters close to other images to avoid destroying a portion of the existing image when the seven by twelve matrix is written. The second application is illustrated in figure A-3.

d. Double Width Flag

If set, causes the displayed characters to be twice as wide as normal as illustrated in figure A-4. The use of double width characters decreases the maximum number of characters per line to 45.

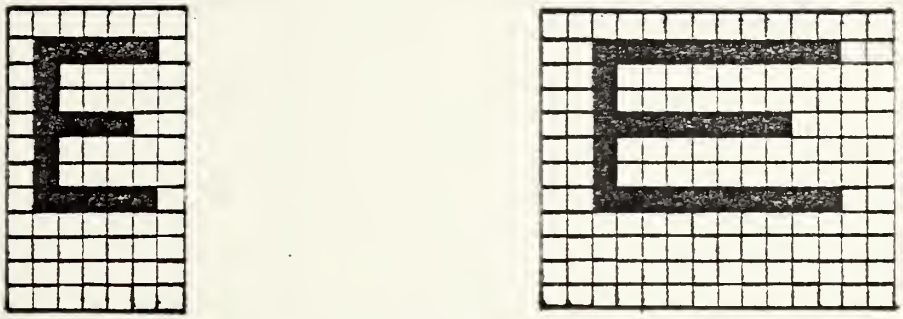
2. Other Instructions

See also strout() and lttr() on Appendix B for additional alphanumeric generating capability.



Effect of Additive Write Flag in Alphanumeric Mode

Figure A-3



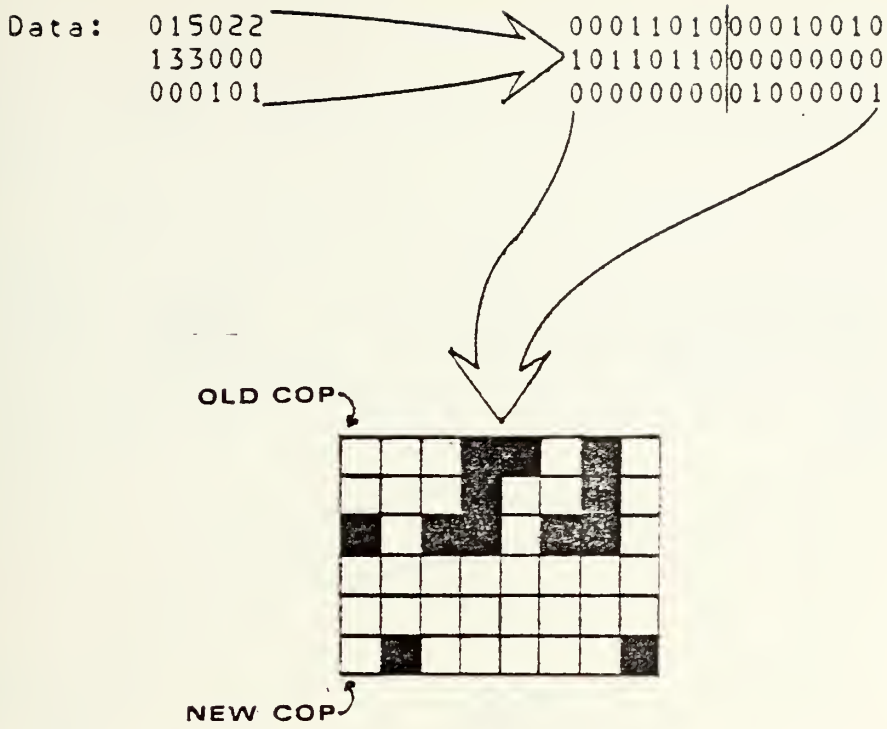
Effect of Double Width Flag in Alphanumeric Mode

Figure A-4

8. TRANSVERSE DATA MODE

All data() instructions issued in transverse data mode are interpreted as passing transverse data. Received data is written in its raw format, that is, each data byte (processed leftmost first) is interpreted as a single bit per pixel description of eight consecutive pixels along a particular real screen line. As illustrated in figure A-5, consecutive data bytes are written from left to right beginning at the COP and continuing down. All subsequent bytes are written beginning at the same element on the next line. The COP after the operation is indicated in figure A-5.

The typical use of transverse data is to define special symbols and characters which can not be found in the standard character set.



Transverse Data Processing

Figure A-5

1. Applicable Control Flags

a. Indexed Addressing Flag

The effect of indexed addressing is as explained above in Virtual Screen Addressing.

b. Reverse Background Flag

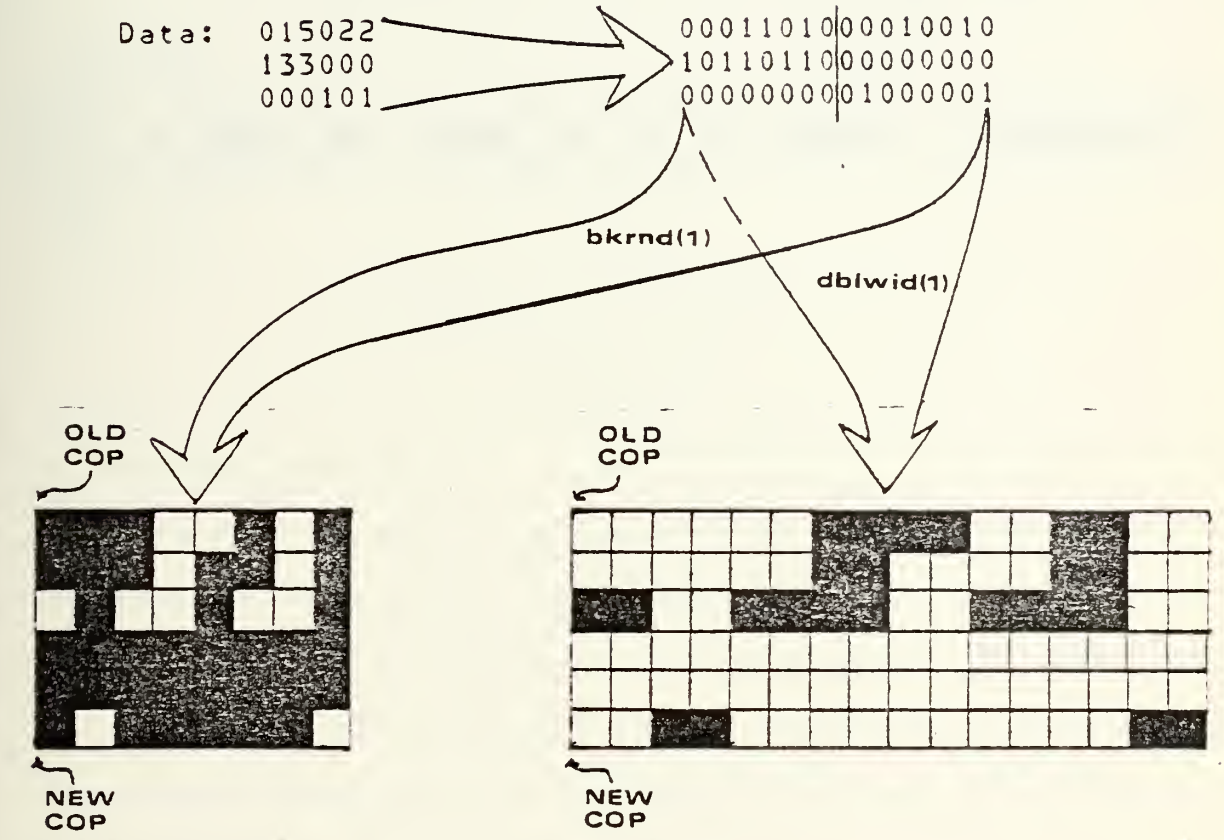
If set, the data is displayed as indicated in figure A-6. In effect, the complement of the data is displayed.

c. Double Width Flag

If set, each bit is reflected in two consecutive pixels instead of one, thereby giving an image sixteen pixels wide instead of eight. The result is illustrated in figure A-6.

d. Additive Write Flag

If set, the transmitted data is written over whatever image happens to be at the same location on the screen.



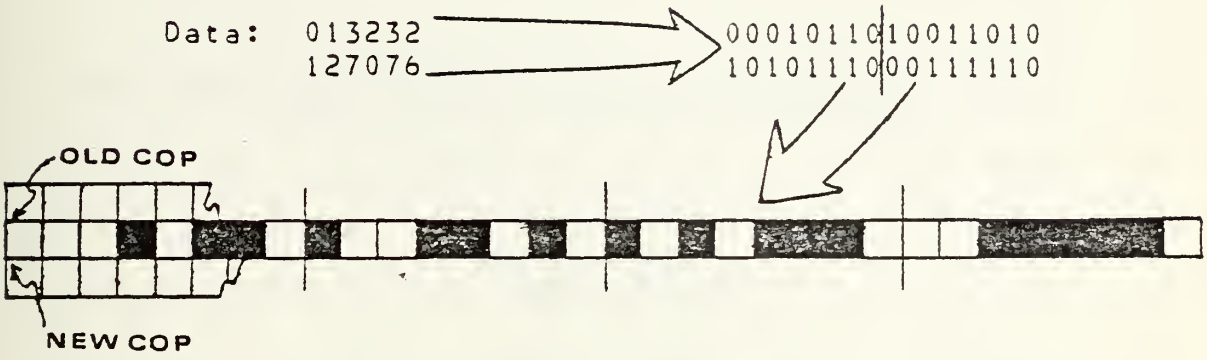
Effect of Reverse Background and Double Width Flags on Transverse Data

Figure A-6

C. RASTER DATA MODE

Raster data mode, except for the direction of the writing process, is identical to the transverse data mode. As illustrated in figure A-7, the consecutive bytes transmitted by the data() instruction are written horizontally from left to right beginning at the COP. All subsequent bytes are written beginning at the next pixel on the same line. The

COP after the operation is as illustrated in figure A-7. The control flags that apply to transverse data mode also apply to raster data mode in the same manner.



Raster Data Processing

Figure A-7

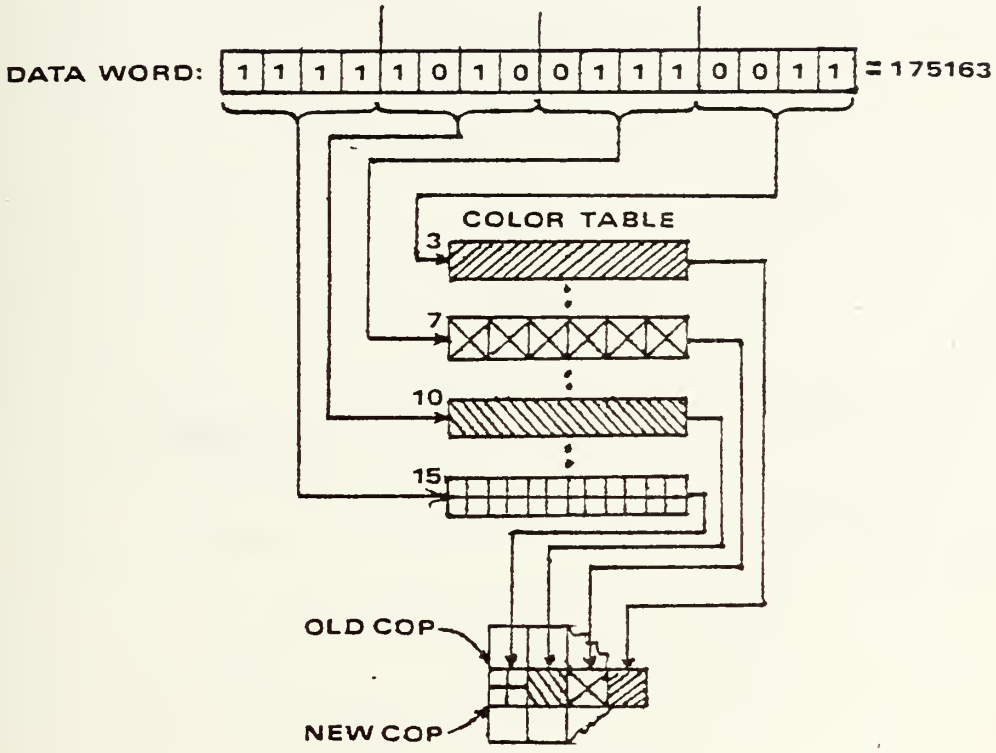
D. COMPLEX DATA MODE

This mode is used to write data on the screen exactly the same way that raster data does. However, a third dimension is added in this mode in that each pixel is described by four bits of the passed data. Therefore, a single data word describes four pixels instead of two as in the previous data modes.

The benefit of this mode is that the color of each pixel being passed is defined by the four bits that are assigned to that pixel. The four bits represent a number from zero through fifteen which is used as a pointer into the color lookup table. If the pointer's value is six (0110), the color of the pixel will be whatever is indicated in entry six of the color table in use.

It can be seen that this mode overrides the color designated by the color() instruction. Interesting images can be achieved by the use of this data mode. All flags applicable

to raster data are applicable to complex data except additive write. Figure A-8 illustrates how a data word is interpreted in this mode.



Complex Data Processing

Figure A-8

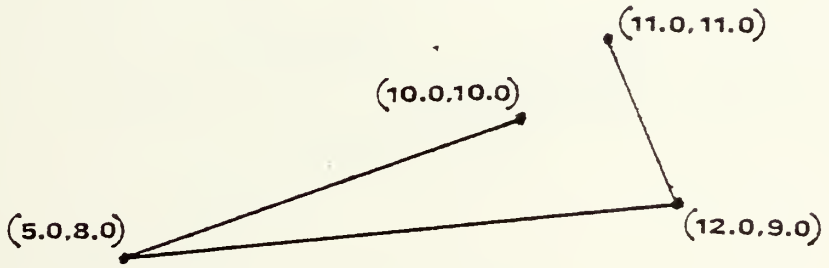
E. GRAPHIC VECTOR MODE

The graphic vector mode draws lines between arbitrary end points. The starting point is defined by the existing COP or can be defined by issuing a strtxy() to establish the new COP. The end point can be defined by issuing either a pointr() instruction or a point() instruction. The first uses relative addressing and the second uses absolute or indexed depending upon the control flags' condition.

After execution of an instruction, the COP is then the end point of the vector just drawn. Therefore, a linked

line, as illustrated in figure A-9, can be drawn by issuing a strtxy() followed by successive point() or pointr() instructions.

```
Instructions:  setmode(4,0);
               strtxy(10.0,10.0);
               point(5.0,8.0);
               point(12.0,9.0);
               pointr(-1.0,2.0);
```



Drawing Vectors in Graphic Vector Mode

Figure A-9

1. Applicable Control Flags

   a. Indexed Addressing Flag

      The effect of indexed addressing is as explained above in Virtual Screen Addressing.

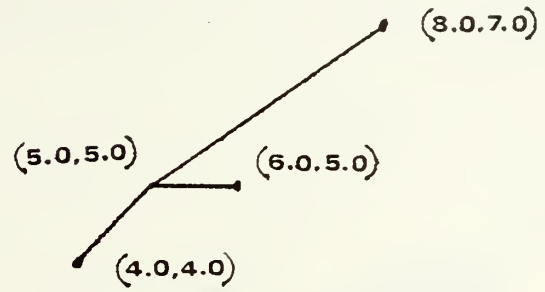
   b. Reverse Background Flag

      If set, the lines are drawn in the background color.

   c. Fixpoint Flag

      If set, causes the point issued by a strtxy() to be the starting point for all subsequent end points issued. Therefore, the lines are drawn with a common point as illustrated in figure A-10.

```
Instructions:  setmode(4,0);
               fixpt(1);
               strtxy(5.0,5.0);
               point(8.0,7.0);
               point(6.0,5.0);
               pointr(-1.0,-1.0);
```



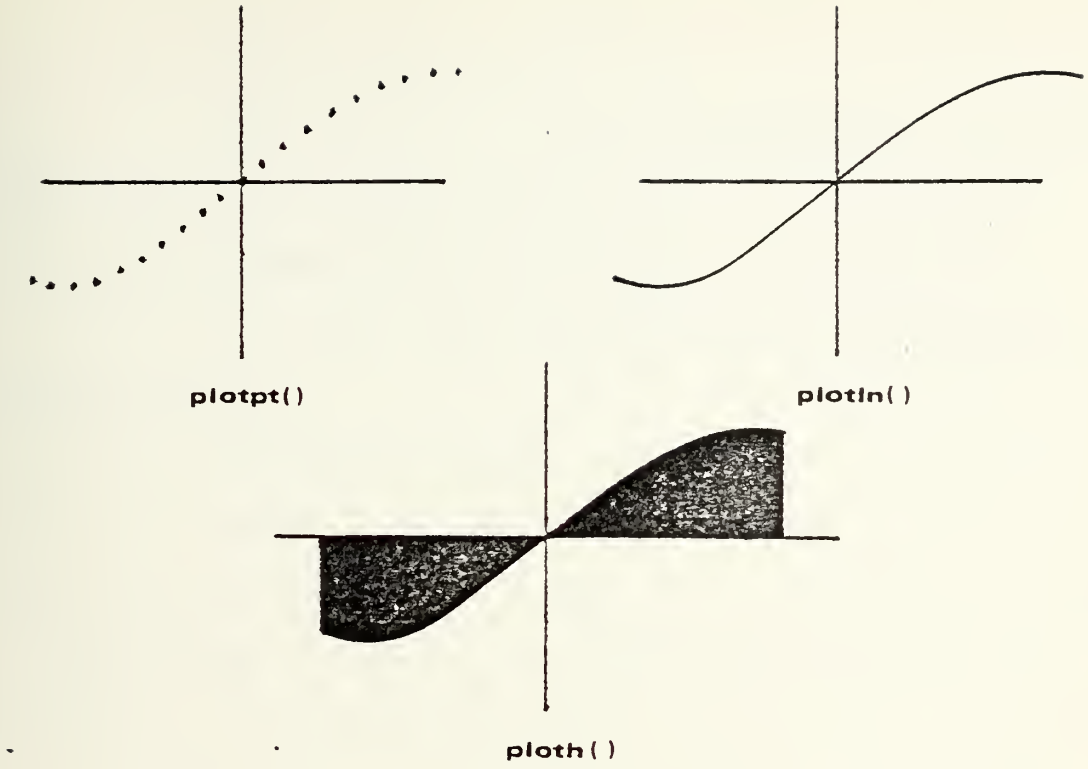
Drawing Vectors with Fixpoint Flag

Figure A-10

F. GRAPHIC PLOT MODE

The functions performed by the graphic plot mode have been implemented as a set of plot routines in the user interface. The three routines are plotpt(), plotln() and ploth().

The first plots a function as a series of points. The second plots a function as a continuous line and the third plots a function in a histogram style plot. Each of the methods requires the user to specify the points to be plotted. The three plotting methods are illustrated in figure A-11. Detailed instructions on their use can be found in Appendix B. It is recommended that these routines be used in lieu of the graphic plot mode as they are simpler to use and provide a more powerful capability. However, for the programmer who is interested in using the assembly language level instructions necessary in this mode, he is referred to reference [2].



Plotting Methods

Figure A-11

G. GRAPHIC CARTESIAN MODE

The graphic cartesian mode draws solid rectangles between arbitrary endpoints. The endpoints define opposite corners of a rectangle. The COP is always the starting point of the drawing unless a new COP is defined with a strtxy() instruction. The second point can be issued with a point() or pointr() instruction. After the operation is performed, the new COP is the second point that was issued.

By issuing successive points, linked rectangles can be drawn as illustrated in figure A-12. In this case, the ending point for one rectangle becomes the starting point for the next.

1. Applicable Control Flags

a. Indexed Addressing Flag

The effect of indexed addressing is as explained above in Virtual Screen Addressing.

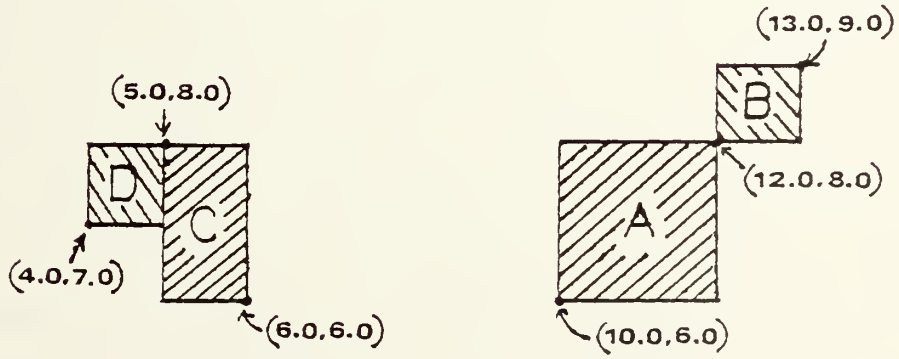
b. Reverse Background Flag

If set, the rectangles are drawn in the background color.

c. Fixed Point Flag

If set, all rectangles are drawn with a common starting point much in the same way it is done in the graphic vector mode. A strtxy() instruction determines the common point, and subsequent point() or pointr() instructions determine the rectangles as illustrated in figure A-12.

```
Instructions:  setmode(6,0);
               strtxy(10.0,6.0);
               point(12.0,8.0);  // draws A.
               point(13.0,9.0);  // draws B.
               fixpt(1);
               strtxy(5.0,8.0);
               point(6.0,6.0);   // draws C.
               point(4.0,7.0);   // draws D.
```



Drawing Rectangles With and Without Fixpoint Flag

Figure A-12

H. GRAPHIC ELEMENT MODE

The graphic element mode draws a single pixel on the screen as determined by a point() instruction or a pointr() instruction. After execution, the COP is on the same real screen line and one screen element to the right.

1. Applicable Control Flags

a. Indexed Addressing Flag

The effect of indexed addressing is as explained above in Virtual Screen Addressing.

b. Reverse Background Flag

If set, the pixels are drawn in the background color.

c. Double Width Flag.

If set, two consecutive pixels on the same line are drawn instead of one.

IV. USING COLOR

As mentioned earlier, all images that are displayed on the screen are displayed according to an entry in the video lookup table which can also be referred to as the color lookup table. This section will discuss how to define colors the user desires, load the color lookup table, use the special interactive routine to build color tables and use the color lookup table in user programs.

A. DEFINING COLORS

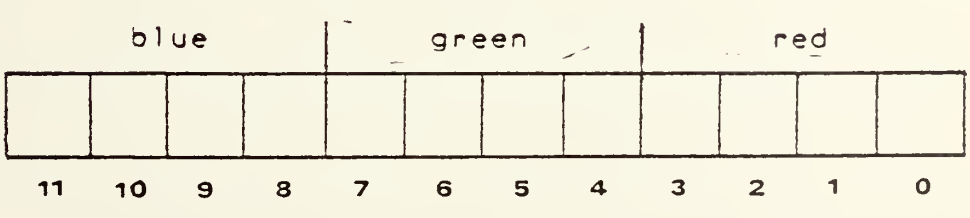
The RAMTEK hardware defines colors by combining red,

green and blue in varying intensities to produce color. Since the color lookup table, from which the CRT receives its instructions on how to mix the three colors, stores information in digital format, the user is limited to 4096 possible color definitions. Since the table has a finite length of sixteen twelve bit entries and only one table can be used at any one time, only sixteen colors can be displayed at any one time on the screen. It must also be kept in mind that entry zero in the table is used as the background color that is to be displayed on the screen thereby leaving fifteen colors for image drawing.

In order to define an entry in the color lookup table, it is first important to examine how the color lookup table stores its information.

1. The Color Lookup Table

There are sixteen entries in the color lookup table. Each entry represents the code for a color in a twelve bit word. As illustrated in figure A-13, each word is broken into three, four bit binary numbers which represent the intensities of blue, green and red to be mixed.



Color Lookup Table Entry Format

Figure A-13

It is convenient to think of the entry in each table as an ordered triple: (0-15,0-15,0-15). This can be extended into a three-dimensional definition of the color space in which the RAMTEK is capable of displaying color. Figure 5 in the earlier section on Application of Color illustrates this concept. The x-axis contains the sixteen possible shades of blue; the y-axis contains the sixteen possible shades of green and the z-axis contains the sixteen possible shades of red. By moving within the confines of the cube, all 4096 colors are defined, and their relationship to each other is illustrated.

2. Defining Color Table Entries

In order to define an entry in the color lookup table, it is necessary to transform from an ordered triple to the twelve bit coded form. This is accomplished by a special routine called triple(b,g,r). The ordered triple (b,g,r) directly corresponds to the color cube location representing the color the user desires. The routine returns the properly coded entry for loading into a color table.

B. LOADING THE COLOR LOOKUP TABLE

The RAMTEK interface system permits the use of eighteen separate color tables. The first four tables (0-3) are reserved system tables and cannot be modified in any way. Some of the other tables already have entries in them for the benefit of the user who wishes to use them, and others are empty.

There are three methods of loading a color lookup table

in the user's program. One, the interactive color routine will be discussed separately. The other two are discussed here. The first is a method for loading an entire color lookup table, and the second is for loading a single entry in a color lookup table.

1. Loading an Entire Table

To load a whole color lookup table requires a series of operations. First, the user needs to declare a sixteen word, linear integer array. Then it is necessary to load each entry of the array from zero to fifteen with the sixteen colors that are desired. In order to accomplish this, it is recommended that the triple() routine be used to code the ordered triples and then assign the returned codes to each entry in the array.

After the array has been loaded, it is then entered into the desired color lookup table by the routine clrtbl(n,name). The parameter 'n' is the number of the color lookup table (4-17) and 'name' is the name of the user array which contains the sixteen coded entries to be loaded.

An example of the code to load a color table is as follows:

```
int a[16];   //declares sixteen word array.
a[0] = triple(0,0,0);     // color black.
a[1] = triple(15,0,0);    // color blue.
a[2] = triple(0,15,0);    // color green.
          .
          .
          .
a[15] = triple(15,15,15); // color white.
clrtbl(5,a);      //loads color lookup table.
```

The color lookup table entries contained in the array 'a' are loaded into color lookup table number five and are then ready for use.

2. Single Entry Loading

It may, at times, be desirable to load only one entry in a given color lookup table. In order to do so, the entry is coded with the triple routine and then loaded in a specific table and entry by using the chnge(numb,entry,cont) routine. For example, to load the color defined by the ordered triple (5,6,10) into entry three of color lookup table ten, the following two instructions are issued:

```
cont = triple(5,6,10);
chnge(10,3,cont);
```

If the change takes place in a color lookup table that is being used for display, the change is reflected on the screen immediately.

C. INTERACTIVE COLOR PROGRAM

The purpose of the interactive program is to allow the user to see and modify color lookup tables while seated at the RAMTEK keyboard. It is recommended that the new user sit down at the RAMTEK and execute inter() in order to get an introduction to the color generating capabilitiesof the RAMTEK. There are two basic modes in which the routine operates. Each of these modes has a series of commands which the user can execute. Appendix B describes the commands for each mode in detail in its description of inter().

1. Paging Mode

The paging mode permits the user to page through the

eighteen possible color lookup tables. Upon execution of inter() the paging mode is automatically entered and a set of tutorial instructions are displayed.

2. Edit Mode

The routine provides five separate methods of editing and modifying color lookup tables and their entries. How to use the methods is described in Appendix B in the description of inter().

a. Table Assignment

Table assignment mode allows the user to copy entries from any table into a designated user table.

b. Octal Assignment

Entries in a designated user table can be defined by entering an ordered octal triple representing the color desired.

c. Combining Assignment

Entries from two color lookup tables can be logically OR'd into a designated user table or two whole tables can be OR'd into the user table.

d. Inverting Tables

This method permits the user to invert a designated table. Entry zero becomes entry fifteen, entry one becomes entry fourteen and so on.

e. Copying Tables

This method permits the user to make copies of any table in the system into a selected user table. Thus, the user can modify a system table without destroying it.

D. PROGRAMMING WITH COLOR

Using color on the RAMTEK is not a difficult process once the user understands color lookup tables and their entries. As mentioned earlier, the system has capability at present of storing up to eighteen color lookup tables. Tables zero through three are reserved system tables leaving tables four through seventeen for the user. The system has some preloaded tables which contain a variety of colors that the user may find useful. If not, they can be destroyed as desired.

It is a good practice for the user to design his programs making use of the inter() routine. By placing the inter() routine inside a loop, the user can modify the table or tables the program uses, run the program and return, if desired, to the inter() routine to change entries in the color lookup tables until the desired colors are achieved. The user should then save the codes for future use.

Issuing a colort(n) instruction in the user program causes the designated color lookup table to be loaded into the RAMTEK hardware video lookup table. All images are then displayed relative to the colors contained in table 'n'. In order to select a specific color to be used, the color(e) instruction is issued. All images drawn subsequent to this instruction will be drawn with the color contained in entry 'e' of the color lookup table last selected.

For example, suppose color table five was the table the user wished to use, and it contained the following entries:

entry zero - black (0,0,0)
entry one - red (0,0,15)

```
entry  two   - blue (15,0,0)
entry three - green (0,15,0)
```

The following instructions would display a red rectangle, a blue line and a green letter 'A' on a black background. Entry zero always contains the background color.

```
        colort(5);          //loads color table five.
        color(1);           //selects entry one (red).
//draw red rectangle.
        setmode(6,0);          //set Graphic Cartesian
                                 mode.
        strtxy(5.0,5.0);
        point(10.0,10.0);
//draw blue line.
        color(2);          //select entry two (blue).
        setmode(4,0);          //set Graphic Vector mode.
        strtxy(15.0,15.0);
        point(22.8,10.5);
//draw green letter 'A'.
        color(3);          //select entry three (green).
        setmode(0,0);          //set Alphanumeric mode.
        strtxy(50.0,25.0);
        data(letter,1);    //letter is an array containing
                           //the ASCII code for an 'A'.
```

If at any time a new color table is entered, all existing images will change color to the entries that are in the new table. Also, if at any time an entry in the table being used is changed, any image on the screen that was drawn with that entry number will change to the new color. If in the preceding example the next instruction loaded a table which contained the color white in entries one and two, and blue in three, the rectangle and line would become white and the letter 'A' would become blue.

V. PROGRAMMING THE RAMTEK

There are two routines which it is necessary to place at the beginning of any C program which desires to utilize the user interface for the RAMTEK.

The ramtek() routine must be the first call made in the user program. It is the initialization routine for the user interface and it opens the RAMTEK device and keyboard.

The erase() routine should follow the ramtek() routine. This ensures that any old images on the screen or in the RAMTEK memory will be removed.

Appendix B contains a complete list of the user routines which will relieve the user of a significant amount of programming effort in many cases. Most user routines return negative values for error conditions or if incorrect parameters are passed. It is therefore advisable to be aware of the values and their meanings to aid in debugging programs.

In order to utilize the interface, the following compile command should be issued:

```
cc filename -lr
```

A. RESERVED WORDS

Table A-II contains a list of reserve words. These words should not be used for user names in any programs that utilize the user interface for the RAMTEK device. Some of the words are user interface routine names and others are variables and subroutine names in the interface software.

TABLE A-II

RESERVED WORDS

ADOFF
adon
ALPHA
axis
BKON
BKOFF
bkrnd
blank
BLK
block
bracket
buff
bytecnt
c
chnge
clrhold
clrtbl
code
codeit
coke
color
colort
colortbl
comb
COMMA
COMPD
conve
convl
copy
CR
data
datao
dblwid
disp
dump
einst
epage1
epage2
epage3
erase
ERS
fixpt
flip
fname
fp
FPOFF
foon
getf
getnum
GRAPHCRT
GRAPHELM
GRAPHVEC
head
headptr
heat
holdx
holdy
index
inptrs
instr
inst1-inst80
inter
int53
int60
itoa
IXOFF
IXON
LCM
lcmhold
LER
LEX
LE1
LE2
LLR
LLX
LL1
LL2
LTA
LTD
lttr
LXD
moreinst
n0-n17
octbl
out
pause
pick
plotpt
plotln
ploth
point
pointr
proc1
proc2
proc3
proc4
proc5
ptrbuff
putup
qptr
qptr1
qt
quest
q1
ramtek
RASTERD
retchar
scissor
SCR
screen
scroll
SDC0
setmode
setup
size
skip
SSCALL
strout
strtxy
systbl
tblwho
TRANSD
triple
upcnt
vector
wait
WDOFF
WDON
writon
xaxis
xmin
xmax
yaxis
yaxisf
ymin
ymax

APPENDIX B: USER INTERFACE ROUTINE DESCRIPTIONS

This Appendix contains the descriptions of the user interface routines in the format that is followed in the documentation of the UNIX operating system in the Naval Postgraduate School Computer Laboratory.

NAME:

axis - draw coordinate axis

SYNOPSIS:

```
axis(x,y)
float x, y;
```

DESCRIPTION:

Draws a cartesian coordinate x and y axis on the screen with center at user-defined screen coordinate (x,y).

The operation is totally independent of any control mode issued previously.

The COP is left at (max x-value,y). Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates no portion of the axis lies on the user-defined screen.

NAME:

bkrnd - change reverse background flag

SYNOPSIS:

```
bkrnd(a)
int a;
```

DESCRIPTION:

If a is zero, the reverse background flag in the control mode is set to zero, i.e. turned off.

If a is equal to one, the reverse background flag in the control mode is set to one, i.e. turned on.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the passed integer a is not equal to zero or one.

NAME:

block - draw solid block

SYNOPSIS:

```
block(x1,y1,x2,y2)
float x1,y1,x2,y2;
```

DESCRIPTION:

A solid block is drawn with opposite corners as defined by user-defined screen coordinates (x1,y1) and (x2,y2).

The COP is left at (x2,y2). The operation is independent of any mode issued previously but is sensitive to all flags applicable to Graphic Cartesian mode.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the block can not be drawn on the user-defined screen.

SEE ALSO:

index(), bkrnd(), dblwid()

NAME:

chnge - change color table entry

SYNOPSIS:

```
chnge(numb,entry,cont)
int numb, entry, cont;
```

DESCRIPTION:

Changes the specified entry in the indicated color table to the passed parameter cont. triple() should be used to put cont into the proper form.

numb is the table number in which the entry is to be changed. entry is the entry number to be changed. cont is the integer twelve bit code which is loaded in the specified entry.

Normal return is zero.

DIAGNOSTICS:

All errors are indicated by negative returned values. The error values and their meanings are as follows:

-1 Indicates numb is less than four or greater than seventeen.

-2 Indicates entry is negative or greater than fifteen.

SEE ALSO:

triple()

NAME:
clrtbl - load color table

SYNOPSIS:
```
clrtbl(n,name)
int n, *name;
```

DESCRIPTION:
Loads color table number n with the codes contained in the array pointed to by name;

The array is an integer array containing sixteen integers which represent the octal code of the color table entries between zero and sixteen. The array is assumed to be declared sixteen words in length.

n is an integer value which ranges from four to seventeen.

The color tables from zero to four are system defined tables. They contain the following:

Table 0 - Fifteen shades of grey.
Table 1 - Fifteen shades of blue.
Table 2 - Fifteen shades of green.
Table 3 - Fifteen shades of red.

If the user desires to modify a system table see inter().

/ Normal return is zero.

DIAGNOSTICS:
Returned -1 indicates the passed integer n is less than four or greater than seventeen.

SEE ALSO:
triple()

NAME:

color - select color

SYNOPSIS:

```
color(s)
int s;
```

DESCRIPTION:

The passed integer s is the number of the entry in the current color table. The color located in that entry will be used for all subsequent entities displayed until a different color is issued.

s is between zero and fifteen.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the passed integer s is negative or greater than fifteen.

SEE ALSO:

chnge()

NAME:
colort - select display color table

SYNOPSIS:
```
colort(t)
int t;
```

DESCRIPTION:
The passed parameter t is the number of the color table that is sent to the ramtek device for display.

t is between zero and seventeen.

Normal return is zero.

DIAGNOSTICS:
Returned -1 indicates the passed integer t is negative or greater than seventeen.

SEE ALSO:
clrtbl()

NAME:

data - display raw data

SYNOPSIS:

```
data(name,l)
int *name, l;
```

DESCRIPTION:

The raw data passed in the linear array pointed to by name is displayed on the ramtek according to the current control mode.

l is the length of the array in bytes!

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the passed integer l is negative or zero.

SEE ALSO:

strout(), lttr()

NAME:

dblwid - change double width flag

SYNOPSIS:

```
dblwid(x)
int x;
```

DESCRIPTION:

If x is equal to zero, the double width flag in the control mode is set to zero, i.e. turned off.

If x is equal to one, the double width flag in the control mode is set to one, i.e. turned on.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the passed integer x is not equal to zero or one.

SEE ALSO:

size()

NAME:
erase - erases the screen

SYNOPSIS:
erase()

DESCRIPTION:
The ramtek screen is erased to the background color of the color table currently being used.

NAME:

fixpt - change fixed point flag in control mode

SYNOPSIS:

```
fixpt(x)
int x;
```

DESCRIPTION:

If x is equal to zero, the fixed point flag in the control mode is set to zero, i.e. turned off.

If x is equal to one, the fixed point flag in the control mode is set to one, i.e. turned on.

Meaningful only in Graphic Vector or Graphic Plot mode. In Graphic Vector mode it causes all subsequent vectors to be drawn with one end at the same point as issued by a subsequent or last previous strtxy(). See the user's manual for fixpt()'s meaning in Graphic Plot.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the passed integer x is not equal to zero or one.

NAME:
getf - read floating point number from ramtek keyboard

SYNOPSIS:
getf()

DESCRIPTION:
Returns a floating point number from the ramtek keyboard. The only numbers recognized are: an optional minus sign followed by a string of digits optionally containing one decimal point, then followed optionally by the letter 'e' followed by a signed integer.

Normal return is the floating point number.

SEE ALSO:
getnum(), retchar()

NAME:

getnum - read number from ramtek keyboard

SYNOPSIS:

```
getnum(base)
int base;
```

DESCRIPTION:

Returns a positive or negative integer number from the ramtek keyboard. Numerals typed up to a comma or a c/r on the keyboard are considered to be the number. The routine will not return any value until a c/r or a comma is typed, nor will it accept characters other than digits from zero to nine when base is equal to ten, or digits from zero to seven in case of base eight.

base is the base of the integer number returned. It is restricted to eight (octal) or ten (decimal).

Normal return is the integer number.

DIAGNOSTICS:

All errors are indicated by negative returned values. The error values and their meanings are as follows:

-1 Indicates a c/r was struck without any previous entries.

-2 Indicates the passed parameter base was not equal to eight or ten.

SEE ALSO:

retchar(), getf()

NAME:

index - select indexing

SYNOPSIS:

```
index(i,x,y)
float x, y;
int i;
```

DESCRIPTION:

If i equals zero, indexing is deselected and the indexing flag in the control mode is turned off. The index registers are then loaded with zero.

If i equals one, indexing is selected and the indexing flag in the control mode is turned on. The index registers are then loaded with the displacements passed in x and y. x is loaded in the x-index register and y is loaded in the y-index register.

x must be no greater than the user-defined screen width. y must be no greater than the user-defined screen height.

Normal return is zero.

DIAGNOSTICS:

All errors are indicated by negative returned values. The error values and their meanings are as follows:

-1 Indicates the passed integer i is not equal to zero or one.

-2 Indicates one of the passed displacements x and y is greater than the screen height or width.

SEE ALSO:

screen()

NAME:
inter - interactive color table modification routine

SYNOPSIS:
inter()

DESCRIPTION:
Sets up and runs a tutorial program that enables the user to look at, select and modify color tables during program execution. There are two basic modes in which the routine operates. Each of these modes has a series of commands which the user can execute.

1. Paging Mode. Upon execution the paging mode is immediately entered and a set of instructions is displayed. The following commands are then acceptable:

'p' - Indicates a particular table number is to be displayed. The number must then be entered between zero and seventeen.

'n' - Increments table number displayed by one.

'b' - Decrements table number displayed by one.

'i' - Displays instructions.

'q' - Quits from the interactive routine.

'e' - Enters the edit mode.

2. Edit Mode. When the edit mode is entered, a set of instructions is displayed listing the commands that can be issued and the edit method they initiate. The commands are:

't' - Enters table assignment method. Permits entries from any table to be copied into the table designated by the user. The table number which is to receive the copied entries must be entered first.

'o' - Enters octal assignment method. Entries are specified in a designated table by entering the ordered triple that defines the color desired. The table number (4-17) in which the entries are to be defined must be entered first.

'c' - Enters combining assignment method. This

method allows the user to logically OR two separate color lookup table entries into a selected user table entry. Whole tables may also be combined into a selected user table. The user table number (4-17) must be entered first.

'f' - Enters inverting tables method. This method inverts the user table designated by the user. Entry zero becomes fifteen, entry one becomes fourteen, etc. The act of entering the user table number executes the inversion.

'm' - Enters copying tables method. After entry of the user table number that is to receive a copied table the number of the table to be copied is entered. The act of entering the second table number causes the copy to take place.

Upon entry, the editing methods display a list of instructions on how the method is to be used and the results obtained. Each of the methods have commands that may be issued. The legal commands for each edit method are summarized below.

- Table Assignment Method.
  - 'n' - Indicates the table number and entry to be copied follows.
  - 'c/r'-Increments through the entries in the receiving user table.
  - 'p' - Displays the color lookup table be- modified. To return from the display, type a 'q'.
  - 'q' - Quits back to edit mode instructions.

- Octal Assignment Method.
  - 'n' - Indicates the octal triple to be entered as the entry follows.
  - 'c/r'-Increments through the entries in the receiving user table.
  - 'p' - Displays the color lookup table being modified. To return from the display, type a 'q'.
  - 'q' - Quits back to edit mode instructions.

- Combining Assignment Method.
  - 'n' - Indicates the two ordered pairs designating the table number and entry number to be OR'd are to follow.
  - 'c/r'-Increments through the entries in the receiving user table.
  - 'p' - Displays the color lookup table being modified. To return from the display, type a 'q'.
  - 'w' - Indicates two whole tables are to be combined (OR'd) and their numbers are to follow.
  - 'q' - Quits back to edit mode instructions.

- Inverting Tables Method.
  - 'p' - Displays the color lookup table that is being or was inverted.
  - 'q' - Quits back to edit mode instructions.

- Copying Tables Method.
  - 'p' - Displays the receiving table.
  - 'q' - Quits back to edit mode instructions.

NAME:

lttr - display single character

SYNOPSIS:

```
lttr(ch,size)
int ch, size;
```

DESCRIPTION:

Will display the character passed with the size indicated at the current operating point.

After display, the current operating point is on the same line and seven actual screen elements to the right of its last position.

ch contains the ascii code for the character to be displayed.

If size is equal to one, the character is displayed in standard size.

If size is equal to two, the character is displayed in double width size.

The operation is independent of any mode issued previously, but is sensitive to the flags applicable to Alphanumeric mode.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the passed integer size was not equal to one or two.

SEE ALSO:

strout(), data(), size(), index(), bkrnd(), writon()

NAME:

ploth - plot data as a histogram

SYNOPSIS:

```
ploth(x,y,n,base)
float *x, *y, base;
int n;
```

DESCRIPTION:

The values y[i] are assumed to be functions of x[i]. As the function is plotted, the area between the curve and the x-axis, as designated by base, is filled in with the color last selected.

That portion of the histogram that lies on the screen will be plotted.

The operation is totally independent of any control mode issued previously.

x and y are pointers to linear arrays.
n is the number of points to be plotted.

Normal return is zero.

DIAGNOSTICS:

All errors are indicated by negative returned values. The error values and their meanings are as follows:

Returned -1 indicates the passed parameter base is not on the user-defined screen.

SEE ALSO:

plotln(), plotpt()

NAME:

plotln - plot data with connected lines

SYNOPSIS:

```
plotln(x,y,n)
float *x, *y;
int n;
```

DESCRIPTION:

The values y[i] are assumed to be functions of x[i]. As the function is plotted, the successive points are connected by straight lines of the color last selected.

That portion of the plotted curve which lies on the screen will be plotted.

The operation is totally independent of any control mode issued previously.

x and y are pointers to linear arrays.
n is the number of points to be plotted.

SEE ALSO:

ploth(), plotpt()

NAME:

plotpt - plot data with points

SYNOPSIS:

```
plotpt(x,y,n)
float *x, *y;
int n;
```

DESCRIPTION:

The values of y[i] are assumed to be functions of x[i]. The function is plotted with dots for each (x[i],y[i]) coordinate. The dots are the color last selected.

If a point does not lie on the user-defined screen, it is not plotted.

The operation is totally independent of any control mode issued previously.

x and y are pointers to linear arrays.
n is the number of points to be plotted.

SEE ALSO:

ploth(), plotln()

NAME:

point - define point

SYNOPSIS:

```
point(x,y)
float x, y;
```

DESCRIPTION:

Defines a point on the user-defined screen. The point must lie within the user-defined screen.

If in Graphic Vector mode, (x,y) defines the endpoint of a vector and causes it to be drawn. The current operating point is then (x,y).

If in the Graphic Element mode, (x,y) defines a single point on the user-defined screen and causes a dot to be drawn there. The current operating point is then (x,y).

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the passed point does not lie on the user-defined screen.

SEE ALSO:

strtxy()

NAME:

pointr - define a point relative

SYNOPSIS:

```
pointr(x,y)
float x, y;
```

DESCRIPTION:

Defines a point on the user-defined screen relative to the last current operating point.

If in Graphic Vector mode a vector from the last current operating point to a point defined by the last current operating point plus the displacements x and y is drawn.

If in Graphic Element mode a dot is drawn at a point defined by the last current operating point plus the displacements x and y.

The COP is left at the calculated location.

SEE ALSO:

strtxy()

NAME:

ramtek - initiates ramtek system

SYNOPSIS:

ramtek()

DESCRIPTION:

Initiates the ramtek system and sets the default conditions as follows:

1 - Initialize the user-defined screen to 0.0 to 100.0 in x and y.
2 - Loads a shades of grey color table (0) in the ramtek.
3 - Selects color fifteen in table zero for display.
4 - Selects Alphanumeric control mode.
5 - Opens the ramtek for reading and writing.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates ramtek device could not be opened.

NAME:
retchar - read a character from ramtek keyboard

SYNOPSIS:
retchar()

DESCRIPTION:
An ascii code representing the typed character is returned in the lower half of an integer.

SEE ALSO:
getnum(), getf()

NAME:

screen - define user screen

SYNOPSIS:

```
screen(x1,y1,x2,y2)
float x1, y1, x2, y2;
```

DESCRIPTION:

Defines a standard cartesian coordinate system of any scale for the user. The point (x1,y1) becomes the coordinate of the lower left corner of the screen. The point (x2,y2) becomes the coordinate of the upper right corner of the screen.

x1 must be strictly less than x2.

y1 must be strictly less than y2.

All subsequent user coordinates are interpreted according to this user-defined screen.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates x1 is not less than x2 or y1 is not less than y2. An error message is also printed on the terminal screen.

NAME:

scroll - scroll screen

SYNOPSIS:

```
scroll(a,cnt)
char a;
float cnt;
```

DESCRIPTION:

The current displayed picture on the screen is scrolled up or down. Information scrolled off the top of the screen will be scrolled in the bottom and vice-versa.

If a is the character 'd', the direction of the scroll will be down.

If a is the character 'u', the direction of the scroll will be up.

cnt is the number of user-defined y-units that the picture is to be scrolled. cnt can be no larger than the screen height.

Normal return is zero.

DIAGNOSTICS:

All errors are indicated by negative returned values. The error values and their meanings are as follows:

-1 Indicates the passed parameter cnt was less than zero or greater than the screen height.

-2 Indicates the passed parameter c was not a 'd' or 'u'.

NAME:

setmode - select control mode

SYNOPSIS:

```
setmode(a,b)
int a, b;
```

DESCRIPTION:

Selects the control mode according to the passed parameters a and b. a represents the control mode as follows:

0 - Alphanumeric
1 - Transverse Data
2 - Raster Data
3 - Complex Data
4 - Graphic Vector
5 - Graphic Plot
6 - Graphic Cartesian
7 - Graphic Element

All flags are turned off if b is equal to zero. If b is equal to one, any flags set in the previous mode are also set with the mode selected by a. All entities displayed subsequent to this call are displayed according to the selected mode. Routines that disregard the selected mode are axis(), block(), inter(), lttr(), ploth(), plotln(), plotpt() and vector().

Normal return is zero.

DIAGNOSTICS:

All errors are indicated by negative returned values. The error values and their meanings are as follows:

-1 Indicates the passed parameter a was less than zero or greater than seven.

-2 Indicates the passed parameter b was not equal to zero or one.

SEE ALSO:

ramtek(), bkrnd(), dblwid(), fixpt(), index(), writon()

NAME:

size - select letter size

SYNOPSIS:

```
size(r)
int r;
```

DESCRIPTION:

If r is equal to one the standard character size is used.

If r is equal to two the double width character size is used.

Sets the double width flag in the control mode. Therefore size takes the same action as dblwid().

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the passed parameter r is not equal to one or two.

SEE ALSO:

dblwid(), strout(), lttr()

NAME:

strout - output character string

SYNOPSIS:

```
strout(sp)
char *sp;
```

DESCRIPTION:

Outputs a character string no greater than 100 characters long beginning at the current operating point and continuing on the same line. After completion, an automatic line feed occurs which defines a new current operating point on the next line at the same starting point as the previous line.

sp points to the character string to be output.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the string contained more than 100 characters. In this case no characters will be displayed.

SEE ALSO:

data(), size(), lttr()

NAME:

strtxy - establish current operating point

SYNOPSIS:

```
strtxy(x,y)
float x, y;
```

DESCRIPTION:

Establishes the current operating point on the screen for subsequent instructions. If the current mode is Graphic Vector with the fixed point flag set, it establishes the base from which the vectors are drawn.

x is a user-defined screen location in x. y is a user-defined screen location in y.

(x,y) must lie within the user-defined screen.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates (x,y) does not lie on the user-defined screen.

SEE ALSO:

fixpt(), screen(), point()

NAME:

tblwho - request for current color table number

SYNOPSIS:

tblwho()

DESCRIPTION:

Returns an integer value which is the table number of the color table that is currently being used for display.

NAME:

triple - code color from three number triple

SYNOPSIS:

```
triple(b,g,r)
int b,g,r;
```

DESCRIPTION:

The three input parameters representing the amounts of blue(b), green(g) and red(r) are coded into an integer which is suitable for insertion into color table entries.

b is between 0 and 15 and represents the amount of blue desired.

g is between 0 and 15 and represents the amount of green desired.

r is between 0 and 15 and represents the amount of red desired.

Normal return is an integer representing the code.

DIAGNOSTICS:

Returned -1 indicates input parameters b, g or r are negative or greater than 15.

SEE ALSO:

chnge(), clrtbl()

NAME:

vector - draw single vector

SYNOPSIS:

```
vector(x1,y1,x2,y2)
float x1, y1, x2, y2;
```

DESCRIPTION:

A vector is drawn on the screen from user-defined screen coordinate (x1,y1) to (x2,y2). The current operating point is left at (x2,y2).

The operation is independent of any mode issued previously but is sensitive to the flags applicable to the Graphic Vector control mode. Undesired results may be obtained if the fixpt flag is set previous to calling vector().

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates no portion of the indicated line lies on the user defined screen.

SEE ALSO:

bkrnd(), dblwid, fixpt(), index()

NAME:

writon - additive write

SYNOPSIS:

```
writon(w)
int w;
```

DESCRIPTION:

If w is a one, the additive write flag in the control mode is turned on causing subsequent entities in Alphanumeric, Raster Data and Transverse Data modes to write on top of previous entities without destroying them.

If w is a zero, the additive write flag in the control mode is turned off.

Normal return is zero.

DIAGNOSTICS:

Returned -1 indicates the passed integer w is negative or greater than one.

REFERENCES

[1] Ritchie, D. M., "C Reference Manual", Bell Telephone Laboratories, Murray Hill, New Jersey 07974, 1975.

[2] "Ramtek GX-100B Programming Manual", Ramtek Corp., 1975.

[3] Wagner, F. V. and LaHood, J., "Computer Graphics - Software Design", p. 99-133, "Computer Graphics", ed. Gruenberger, F., Thompson Book Company, 1967.

[4] Sutherland, I. E., "SKETCHPAD - A Man-Machine Graphical Communication System", AFIPS Conf. Proc., Vol. 23, p. 329-346, SJCC 1963.

[5] Knowlton, K.C., "A Computer Technique for Producing Animated Movies", AFIPS Conf. Proc., Vol.25, p. 57-85, SJCC 1964.

[6] Teitleman, W., "Real Time Recognition of Hand-Drawn Characters", AFIPS Conf. Proc., Vol. 26, p. 559-575, SJCC 1964.

[7] Jacks, E. L., "A Laboratory for the Study of Graphical Man-Machine Communication", AFIPS Conf. Proc., Vol 26, p. 343-350, SJCC 1964.

[8] Newman, W. M. and Sproull, R. F., "Principles of Interactive Computer Graphics", p. 359, McGraw Hill, 1973.

[9] "Ramtek GX-100A Programming Manual", Ramtek Corp., 1974.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexanderia, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4.	Professor George A. Rahe, code 52Ra Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5.	LTJG Gary M. Raetz, USN, Code 52Rr Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
6.	Commanding Officer Naval Electronic Systems Command Code 320 Washington, D. C. 20360 Attn: CDR Miller	1
7.	1/Lt. Roger L. Nesslage, USMC 12709 Colby Drive Woodbridge, Virginia 22192	1

Thesis 165875
N415 Nesslage
c.1 The design of a user interface for a color, raster scan graphics device.